SYSTEMS AND METHODS FOR DETERMINING THE PHYSICAL LOCATION OF A COMPUTER'S NETWORK INTERFACES

RELATED APPLICATION

This application is a continuation in part of application serial number 09/557,498, filed on April 24, 2000.

TECHNICAL FIELD

This invention relates generally to computer networks and, more specifically, to a system for determining where a computer's connections to networks are physically located.

BACKGROUND OF THE INVENTION

Computers today move around and connect to and disconnect from networks much more frequently than they did in the past. Laptops move around among one or more office networks and the home network. As a mobile computer with a radio network link passes from one radio access point to another, its network connection is, in effect, dropped and then reinstated.

Mobility causes problems, however, when a computer either does not know where it is located or does not know the locations of resources of interest to it. Mobile computers, applications, and system services cannot adjust their behavior based on their physical locations if they do not know their locations. Even static computers could use information about the physical locations of themselves and of the machines with which they are communicating. For example, a desktop computer that is unaware of its physical location cannot adequately choose the proxy servers to which it connects, set the time zone value for its clock, choose a nearby printer as the system default, or choose which virtual private networks to join. A server that is ignorant of the physical locations of its remote clients cannot measure the geographic spread of interest in its services, restrict content, or assess shipping charges.

United States Patent Application "Systems and Methods for Uniquely and Persistently Identifying Networks" describes a service that discovers and presents to applications the identities of the logical networks to which their host computer is attached. While helpful in the above scenarios, this information is sometimes not sufficient. Only with relatively precise geographic information can some applications and

services adequately conform their behavior to account for the realities of their location. These applications are left having to discover geographic information themselves before they can respond appropriately. The problem is especially acute for mobile computers and for servers with a geographically dispersed, and possibly mobile, set of clients because they must constantly seek location information about themselves or about others.

Forcing each application and system service to discover geographical information is complicated by the fact that multiple methods are often available, depending upon the types of the networks and the machines on them. Some methods currently in use include the IP address of the interface, the subnetwork address, the Domain Name System name, the media access control address of a network interface card, the wireless network name, among others. These methods present problems to the applications because the methods vary in their availability for a particular use, in their reliability, in the precision of the location information they provide, and in the format of their results.

SUMMARY OF THE INVENTION

The above problems and shortcomings, and others, are addressed by the present invention, which can be understood by referring to the specification, drawings, and claims. The present invention uses various methods for discovering physical location information. The invention decides which method or methods are applicable to each network interface on the computer, applies those methods, and collects the results. The results are converted into a common format, such as latitude and longitude, and keyed to the names of the logical networks accessible through the network interfaces. The information is made available to whatever system services or applications request it.

In addition to providing physical location information, some of the methods used provide estimates of the quality and reliability of the information, such as error ranges and confidence intervals. The invention collects this error information when it is available. If more than one method is available for use on a network interface, the invention runs all of them and presents the information from each method, sorted by the size of the information's error range, if known.

Clients of the physical location information are notified when the information provided to them changes or when new information becomes available. Clients may

specify a threshold so that location changes of a magnitude below the threshold are not reported to them.

BRIEF DESCRIPTION OF THE DRAWINGS

While the appended claims set forth the features of the present invention with particularity, the invention, together with its objects and advantages, may be best understood from the following detailed description taken in conjunction with the accompanying drawings of which:

Figure 1 shows a community of computer systems connected by multiple networks;

Figure 2 is a block diagram generally illustrating an exemplary computer system that supports the present invention;

Figure 3 presents the International Standards Organization Open Systems Interconnection (ISO/OSI) model for hierarchically-layered network communications protocols;

Figure 4A is a block diagram showing some of the functions involved in running a hierarchically-layered network communications protocol on a computer system;

Figure 4B is a block diagram presenting one possible implementation of the functions described with respect to Figure 4A;

Figure 5 is a flow diagram showing how one embodiment of the present invention provides network connection information to applications;

Figure 6A shows a laptop PC and three locations where the laptop's owner may wish to use it;

Figure 6B shows, for each of the three locations of Figure 6A, the network information provided and the configuration a particular application may choose when connected to a logical network at that location;

Figures 7A and 7B show a virtual private network connection from a user's home to her office and the information provided about this configuration;

Figures 8A through 8C show two computing devices using the Network Address Translator to share a connection to the Internet and the information provided about this configuration;

Figure 9 presents a circumstance in which some embodiments of the invention may return the same name for different networks;

Figures 10A and 10B present a situation in which some embodiments of the invention may return different names for the same network;

Figure 11A is a network diagram showing a computing device connected to four different types of networks;

Figure 11B is a flowchart showing how to determine the connectivity types of the networks of Figure 11A;

Figure 11C is a flowchart showing how to determine whether a network has connectivity to the Internet;

Figure 11D is a block diagram showing network connectivity type and Internet connectivity information for the four networks of Figure 11A;

Figures 12A and 12B show a virtual private network connection from a user's home to her office and one embodiment of the physical location records generated in this scenario; and

Figures 13A through 13D are flow diagrams illustrating the actions performed by one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

Turning to the drawings, wherein like reference numerals refer to like elements, the invention is illustrated as being implemented in a suitable computing environment. The following description is based on exemplary embodiments of the invention and should not be taken as limiting the invention in any way. Sections I through IV describe media sense awareness and notifications of changes in network connectivity. These sections are based on United States Patent Application Serial Number 09/652,501, "Systems and Methods for Resynchronization and Notification in Response to Network Media Events." Sections V and VI describe a service that identifies the logical networks to which a computer is attached. These two sections are based on United States Patent Application "Systems and Methods for Uniquely and Persistently Identifying Networks." Sections VII and VIII describe an exemplary embodiment of the invention that determines where a computer's network interfaces are physically located.

In the description that follows, the invention is described with reference to acts and symbolic representations of operations that are performed by one or more computers, unless indicated otherwise. As such, it is understood that such acts and operations, which are at times referred to as being computer-executed, include the manipulation by the processing unit of the computer of electrical signals representing data in a structured form. This manipulation transforms the data or maintains them at locations in the memory system of the computer, which reconfigures or otherwise alters the operation of the computer in a manner well understood by those skilled in the art. The data structures where data are maintained are physical locations of the memory that have particular properties defined by the format of the data. However, while the invention is being described in the foregoing context, it is not meant to be limiting as those of skill in the art will appreciate that various of the acts and operations described hereinafter may also be implemented in hardware.

I. Sensing Network Media Events

Sections following this one discuss how computer systems may respond when they sense network media events. This section provides information about the events themselves.

In Figure 1, several computer systems with different communications mechanisms are configured to communicate with one another. While Figure 1 shows all of the computing devices as resembling general-purpose computers, they may in practice range from multiprocessor mainframes to personal digital assistants, telephones, set top boxes for televisions, etc. A LAN 106 connects a first computing device 100, a server 102, and a router 104. The router also has a communications link to an internetwork 108 which may be a corporate intranet or the Internet. A second computing device 110 connects to the internetwork 108 by way of a dial-up modem 112 and telephone line. A third computing device 114 uses wireless technology to communicate. The wireless technology may be, for example, infrared or optical, but this example portrays a radio connection. Two radio access points (RAPs), 116 and 118, are connected to the internetwork 108 and may enable the third computing device 114 to communicate with the other computing devices connected to the internetwork. The first computing device 100 can also communicate via wireless means, in addition to using the LAN 106.

Figure 2 is a block diagram generally illustrating an exemplary computer system that supports the present invention. The computing device 100 is only one example of a suitable environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing device 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in Figure 2. The invention is operational with numerous other general-purpose or special-purpose computing environments or configurations. Examples of well-known computing systems, environments, and configurations suitable for use with the invention include, but are not limited to, personal computers, servers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set-top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, and distributed computing environments that include any of the above systems or devices. In its most basic configuration, computing device 100 typically includes at least one processing unit 200 and memory 202. The memory 202 may be volatile (such as RAM), non-volatile (such as ROM, flash memory, etc.), or some combination of the two. This most basic configuration is illustrated in Figure 2 by the dashed line 204. The computing device may have additional features and functionality. For example, computing device 100 may include additional storage (removable and non-removable) including, but not limited to, magnetic and optical disks and tape. Such additional storage is illustrated in Figure 2 by removable storage 206 and non-removable storage 208. Computer-storage media include volatile and non-volatile, removable and non-removable, media implemented in any method or technology for storage of information such as computer-readable instructions, data structures, program modules, or other data. Memory 202, removable storage 206, and non-removable storage 208 are all examples of computer-storage media. Computer-storage media include, but are not limited to, RAM, ROM, EEPROM, flash memory, other memory technology, CD-ROM, digital versatile disks (DVD), other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage, other magnetic storage devices, and any other media which can be used to store the desired information and which can accessed by computing device 100. Any such computer storage media may be part of the computing device. The device 100 may also contain communications connections 210 that allow the device to communicate

with other devices. Communications connections 210 are examples of communications media. Communications media typically embody computer-readable instructions, data structures, program modules, or other data in a modulated data signal such as a carrier wave or other transport mechanism and include any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communications media include wired media, such as wired networks (including the LAN 106 of Figure 1) and direct-wired connections, and wireless media such as acoustic, RF, infrared, and other wireless media. The term computer-readable media as used herein includes both storage media and communications media. The computing device 100 may also have input devices 212 such as a keyboard, mouse, pen, voice-input device, touch-input device, etc. Output devices 214 such as a display, speakers, printer, etc., may also be included. All these devices are well know in the art and need not be discussed at length here.

Network logic within a computing device constantly monitors the characteristics of the physical medium that links the device to a network. Very generally, when that logic senses a change in the medium's characteristics, the logic reacts, possibly changing the way the device connects to the network. Also, the network logic informs applications running on the device of the change so that they may react appropriately. The following examples of network media events will aid in understanding the teachings of the present invention. Sections following this one describe in detail responses to the sensed change, both from the network logic and from the applications.

In a first example, assume that the computing device 100 is monitoring its link to the LAN 106 for the most fundamental of network media events: a media disconnect. In a media disconnect event, all incoming and outgoing traffic fails on the link. The manner in which a disconnect event is sensed may vary with the type of link. On some links, the computing device may sense the disconnect immediately while in others, the computing device may not know until a certain amount of time passes without detectable activity on the link. In any case, the response is usually the same with the computing device first attempting to reconnect to the network over the link. Failing that, the computing device might attempt to communicate over another link. This particular computing device may

try the latter strategy because it has a radio transmitter. With the LAN inaccessible, the computing device may be able to reroute its traffic to RAP A 116 and then on to the other devices.

The complement of a media disconnect event is a media connect event. The computing device 100 may note that its connection to the LAN 106 is now working again. If that connection is faster or cheaper than the computing device's radio link, the computing device 100 may reroute its traffic off the radio link and back onto the LAN. In one embodiment of the present invention, the computing device 100 keeps track of how long the link was disconnected. If the disconnect period was very short, then the computing device proceeds to use the link as if nothing happened. For longer disconnects, however, procedures running on the computing device or on other devices may have timed out. In such a situation, the computing device needs to reestablish the network link as if it were connecting for the first time. Devices that are not always connected to a network, such as the computing device 110 that dials in, must go through this new connection routine whenever they access a network. To complicate matters, the computing device cannot always assume that the network to which it is now connected is the same network to which it was connected before the media disconnect event. In one embodiment of the present invention, the computing device performs some type of discovery to learn the nature of the newly connected network.

Wireless complicates these simple scenarios of media disconnect and connect events. Assume that the computing device 114 is using its transmitter to access devices via RAP A 116. Assume also that the computing device 114 is moving away from RAP A toward RAP B 118. While moving, the computing device monitors the strength of the signals coming from the two RAPs. The strength of RAP A's signal could suddenly go to zero which would be analogous to the LAN media disconnect event discussed above, but it is much more likely that the signal will gradually fade. Sensing the fade, the computing device may try to connect to RAP B even while the signal from RAP A is still tolerable. Thus, a wireless link is susceptible to more than simple media disconnect and connect events because the wireless link is not just "up" or "down" but may be "better" or "worse" or "mostly the same" as one or more other available links. A wireless computing

device may need to closely monitor all available links in order to make intelligent transmission choices.

This radio scenario hints at more complications of wireless communications. A wired device usually knows how many connections it can have: at most, one per network interface. A wireless device, on the other hand, has a potential connection to each access point whose signal strength is above a given threshold and these access points may appear and disappear unpredictably. This exacerbates the problem mentioned above in relation to wired networks: a connect following a disconnect need not be to the same network. In one embodiment of the invention, wireless links are checked to see to what they are connecting.

Going beyond media disconnect and connect events, network logic may detect a change in the error rate of data transmitted over the link. This may be due to link degradation, such as when a standard telephone line gets wet, when a radio access point moves farther away, or when a radio link suffers from increased interference. The increased error rate may also be caused by a defective network interface card connected to the network, or may simply be a result of increased traffic loads on the network. If the detected error rate exceeds a certain amount, then the network logic may look to use another network link, even though the current link is not in any sense disconnected.

The effective data rate that a computing device gets from a network link may decrease, independently of any increase in the error rate on the link. This may be due to increased use by others of a shared link or by link logic changing the link's transmission characteristics to accommodate changes in its operating environment. The response could well be the same as for an increased error rate: the network logic could search for another link.

Devices may want to respond to changing conditions before they result in actual errors. For example, network logic may sense that the response time on a network link is increasing substantially, although it is still within the parameters formally defined as acceptable for this type of network link. By definition, this is not an error condition, but the network logic may still use this information in considering whether to send traffic by a different route.

Other types of media events can be sensed including a change in the latency experienced by packets traversing the network, a change in the variation in latency (jitter), a change in the maximum packet size allowed on the network, etc. Responses can be developed for many of these events.

The above examples begin to show the richness of the information available to a device that is critically aware of the state of its network media. The following sections build on this richness to show some of things that can be done with this information.

II. An Implementation of the Network Logic

Before detailing how the network logic responds to network media events that it senses, it may help to have in mind one particular embodiment of that network logic. On each computing device in Figure 1, network logic implements the communications protocols used. Many, but not quite all, of today's communications protocols follow the International Standards Organization Open Systems Interconnection (ISO/OSI) protocol model shown in Figure 3. In this model, the overall task of enabling network communications is divided into subtasks and those subtasks are each assigned to a logical layer in the protocol stack. The stack is hierarchical: each layer has a defined interface with the layers above and below it. Logically, each layer communicates with its peer layer on another computer, provides services to the layer above it in the stack, and uses the services provided by the layer below it. Physically, data flow down the stack from their originator until the physical layer 300 actually transmits them across the medium of the network connection, shown here as a LAN 106. When the data are received by the target computer, they are passed up the stack with each layer stripping off and responding to the data meant for it while passing the rest of the data up to the next level. For example, the network layer 304 defines how data are routed among networks. The network layer on one computer logically speaks with the network layer on another computer by passing a packet of data down to the data layer 302 on its own computer. The data layer in turn adds a header to the network layer's packet thus creating a data frame which it passes to the physical layer. The physical layer uses the physical network medium to transmit that data frame. When the data frame is received by the target computer, the target's data layer reads the data frame, stripping from it the header information meant for its own use. Then the data layer takes the rest of the frame, consisting of the originator's data packet, and

sends it up to the network layer. Thus, the network layer on the target reads the data as sent by the network layer on the originator without having to decode the data layer header and other information used by the lower layers to transmit the data packet.

The primary advantage of this scheme is that application programs 306 can communicate with each other without concerning themselves with the myriad details of establishing and maintaining a network connection. Protocol layers find the remote application programs with which the originating application program wishes to speak, retransmit information damaged during transmission, ensure that information is received in order, reroute information when communications links fail, and do all the other tasks necessary to present as flawless as possible a connection to the application programs.

It should be remembered that the ISO/OSI protocol stack is a conceptual model only and that no protocol follows it exactly. However, many popular protocols in use today follow this model to a greater or lesser extent and the model makes discussion of the communications tasks it defines more easily comprehendible.

The ISO/OSI model does not, however, specify how a computing device 100 should internally implement the tasks required to support an ISO/OSI layered communications protocol. Figure 4A shows one possible implementation of a computing device supporting the popular Transmission Control Protocol/Internet Protocol (TCP/IP) stack. In this embodiment, but not necessarily in all embodiments, communications flow up and down a stack of computing components that is closely analogous to the layered stack in the ISO/OSI model. Network communications services are presented to the application program 306 by the socket layer 400. The socket layer insulates the application program from the details of the ISO/OSI communications protocol. This insulation is especially valuable when a computing device is connected to more than one network and running more than one communications protocol. (For an example of this, see Figure 4B.) When the application program calls a routine in the socket layer to send data to an application program on another computing device, the request works its way down the stack of protocol components with each component communicating with its peer on the other computing device as per the ISO/OSI model. Even in this embodiment, however, some components do not map directly to ISO/OSI layers. Some protocols do not implement all of the ISO/OSI layers, as shown here by the lack of a specific layer 5

(Session Layer) component. An implementation may also combine the functionality of several ISO/OSI layers into one component. In Figure 4A, layers 3 (Network Layer) and 4 (Transport Layer) are supported by a combined TCP/IP driver 406. A complicated protocol such as TCP/IP requires functions beyond the simple transport of data. In Figure 4A, these functions are represented by the 802.1X component 402, which provides authentication services, and by the dynamic host configuration protocol (DHCP) service 404, which provides for non-static network addresses.

Figure 4B is a more specific version of Figure 4A and illustrates an embodiment of network communications tasks within Microsoft's "WINDOWS" operating system. This embodiment is useful for illustrating how the network logic responds to sensed media events, as described in the next section. For the moment, note that there is often more than one communications task at a given layer. ISO/OSI's physical layer 300 is divided in this implementation between connection-oriented physical layer drivers 414, such as for the Asynchronous Transfer Mode 416, and connectionless physical layer drivers 418, such as for Ethernet 420. Again, one computer may simultaneously implement two sets of ISO/OSI levels 3 and 4, once as TCP running over IP 406, and once as the Appletalk protocol 422. This redundancy gives to the network logic the flexibility to reroute information from one protocol stack to another when it senses a disruption in the service provided by one of the stacks.

III. Actions of the Network Logic in Response to Network Media Events

According to one embodiment of the present invention, having sensed one of the several types of network media events discussed above, the network logic analyzes that event and responds in a manner calculated to best preserve effective communications. The response varies not only with the event, but with the existing communications environment, with the computing device on which the network logic is running, and with the specific implementation of the network logic. This section details responses to some events, taking examples from one possible implementation of the network logic. For purposes of illustration, the computing device is taken to be the general-purpose computer 100 of Figure 1, connected to a TCP/IP network 106. In general, however, the computing device may be any device connected to any communications link, such as a networked printer or even a telephone, etc.

On a Media Disconnect Event

In a preferred embodiment, when a physical layer driver (418 in Figure 4B) detects that its network media has been disconnected, it passes a notification of the event up the protocol stack to the Network Device Interface Specification (NDIS) 424. NDIS uses the IPStatus callback function to pass the notification to the TCP/IP driver 406.

At this point, the TCP/IP driver 406 may choose to wait for a short time and then query the lower protocol levels for the status of the network connection media. If the problem has been cleared up, then the TCP/IP driver need do nothing more. This strategy smoothes out transient problems without having to inform the application programs 306.

If a computing device is equipped with more than one network interface (as is the case, for example, with the first computing device 100 in Figure 1), then the network logic attempts to reroute traffic intended for the failed network interface 420 to another network interface. In one embodiment of the present invention, this reroute is done entirely within the protocol stack, transparently to the application programs 306. Note, however, that even if traffic can be rerouted, the TCP/IP driver 406 may choose to remove the address (or mark it unusable) and give an "IP address deletion" notification. This allows media sense aware applications to synchronize to the new state, if necessary. For example, routing software can respond to an immediate notification of a media disconnect or connect event. The routing software can then use its routing protocol to tell other routers about the event.

If a second network interface is available, the network logic may need to perform one or more of the following operations, in addition to rerouting traffic.

- Automatic Virtual Private Network Failover. If a tunnel to a server was set up over the failed network interface 420, a new tunnel may be set up over the second network interface. Even if no tunnel was set up on the failed network interface, a new tunnel may have to be set up if the second network interface connects to a network different from that of the failed network interface. In either case, tunnel setup requires more than a simple change to the routing table. The tunneling client software must run the tunnel setup logic with the same (if accessible and preferable) or with another tunnel server. The tunneling client software may be

configured with the name or address of the other tunnel server, or it may query the Domain Name System (DNS) over the second network interface.

- Automatic Address Failover. If the second network interface connects to the same network as did the failed network interface 420, then the second interface can move from being passive (connected to the network but with no address) to being active by taking over the address of the failed network interface. Even if the second interface was active before the failure, it can still take the address of the failed interface. This makes the disconnect event transparent to the end stations communicating with this computing device via the failed-over TCP/IP address.
- Binding Update. The computing device 100 can send an IPv6 binding update message to tell a remote computer 110 that the TCP/IP address associated with the TCP connect message is now connected to the second IP address rather than to the failed address.

If the problem persists and traffic cannot be transparently rerouted, then the TCP/IP driver 406 removes the TCP/IP address associated with the failed network interface 420 and passes an "IP address deletion" notification up the stack. Application programs 306 receive that notification through the Transport Driver Interface 426, IPHLPAPI callbacks, the Winsock API (Application Program Interface), or Microsoft's Windows Management Instrumentation. The notification allows users of the TCP/IP protocol stack to operate in the "network unreachable" mode.

The next section describes other responses taken by application programs when they receive this notification.

On a Media Connect Event

Notice of the event is passed up the stack, just as in the case of a media disconnect event, described immediately above. The TCP/IP driver 406 passes the notice up to the DHCP service 404. The DHCP service requests either a DHCP renew or a DHCP discover, the choice depending on whether the service has a non-expired or an expired lease at the time of the media connect event. The TCP/IP driver gets an IP address and other network configuration information (which may either be a renewal of the configuration existing prior to a media disconnect event or may be new). The TCP/IP driver then passes up the stack a "something changed" notification. Higher-level protocol

drivers respond to this notification by tuning their behavior to use the new network connection in the most optimal manner possible. Some application programs 306 may not recognize the notification; they might have to be restarted to ensure that they use the new network configuration.

If the computing device 100 were disconnected from one network and then connected to another, its DHCP renew or discover request would be answered with a NACK from the new network's DHCP server. The computing device then gets a new address and configuration from the new network's DHCP server.

If the computing device has a statically configured address, then there is normally no need for a DHCP renewal or discovery. The static address is simply replumbed to make the protocol stack operational on the newly connected network interface. However, an Address Request Protocol message is sent to detect whether the static address is a duplicate of an address already in use. In the case of a disconnect from one network followed by a connect to another, the computing device can discover to which network it is connected by sending a DHCP discover. If the computing device determines that it is on a different network from the one for which it is configured, it can decide to stay in the disconnected state, go into the auto-net state in order to gain connectivity to other auto-net machines, or simply take the returned DHCP address if the DHCP server is available. A stored policy can tell the computing device which action to take in this situation.

Once the computing device has discovered and plumbed its new network configuration, it can proceed to do the following. The computing device rejoins the IP multicast groups it was in before the media disconnect event. There are two types of groups to be joined—permanent and temporary. The "all hosts multicast" and "all routers multicast" groups are examples of permanent groups that all hosts and all routers, respectively, must join. Temporary groups are those that a computing device joins as a result of a request from an application program. On a media connect event, the computing device automatically joins the permanent groups when the protocol stack is initialized as part of the address plumbing. Media sense aware application programs can then request that the computing device rejoin its temporary groups.

If IPsec policies were in effect on the computing device, then they should be plumbed correctly. For example, the IPsec policy:

from/to me to/from <IP addresses or wild card>, use

3DES encryption and MD5 authentication

should be plumbed. Note that if the IP address changes as a result of the media connect event, "**me**" in this IPsec policy would expand to the new IP address. This ensures that no information leaks as a result of the change from the former network to the new.

If a quality of service policy of the type:

from me to <IP address>, use xx bits/second

were in effect on the computing device, then the computing device sends RSVP requests for service guarantees of bandwidth on the network and on the destination. This is done even if the computing device were disconnected and then connected back to the same network because the state kept by network elements such as routers may have timed out between the two media events.

If the computing device is "ICMP router discovery enabled," then it sends router discovery requests in order to discover the routers on the network. Similarly, the computing device discovers if other servers and services are available on the new network.

On a Change in Carrier Strength Event

As discussed above, a wireless computing device 114 moving away from one RAP 116 toward another RAP 118 will probably experience a degradation in the signal strength of the former RAP while the strength of the latter's signal increases. The computing device need not wait for an actual media disconnect on the RAP it is using but can anticipate it and respond by scanning for other RAPs. Upon notification of a change in signal strength, the computing device decides whether it is appropriate to connect to a new RAP. If so, it makes the connection and triggers a media connect event specifying the new RAP. Higher-level protocol drivers and applications are notified that the connection was not lost but may have changed. In addition, the following actions are taken:

- On choosing to connect to a new RAP (a virtual media connect event), the computing device sends three Extensible Authentication Protocol (EAP) start authentication messages to the new RAP to determine whether it supports authentication, such as by use of the IEEE 802.1X protocol. If so, then the

computing device completes the authentication process, else it assumes that it is connected to an unauthenticated network.

- Once the computing device is authenticated (or decides it is on an unauthenticated network), it starts the DHCP protocol to get its address and network configuration.
- Subsequent steps are the same as explained above for real media connect events.

On Other Network Media Events

Other network media events described above in the section entitled "Sensing Network Media Events" include a change in the error rate on the communications link, a change in the effective throughput on the link, and a change in the response time of some other device on the network. The same principles described above can be used to enable the network logic to respond to these other network media events.

IV. Actions of Applications upon Notification that Something Changed

As described above, sometimes the network logic can respond to a network media event in such a manner that the application programs need never know of the event. More generally, however, the application programs are informed that something changed so that they may try to alter their operations in response to the changed circumstances caused by the event.

On a Network Changed Event

This is a "generic" event, indicating that something has changed but not saying what. Upon receiving notification of this event, the application enquires to determine what has changed. The most common possibilities are described below.

Media Disconnect

An application program may choose to close down if it cannot operate without network access. On the other hand, some application programs may continue to perform as best as they can while waiting to take advantage of a possible reconnect event in the near future. These applications inform their users of the problem and, if the original network connection cannot be reestablished, they remain ready to take advantage of a new network connection if one becomes available.

Some application programs may not recognize the notification of a media disconnect and so cannot immediately respond to the new circumstances. However, as

soon as they attempt an operation on the failed network connection, they will receive a "network disconnected or unreachable" error and can then respond appropriately.

Media Connect

If there were no working network connections before this event, then the application program changes its state from "network unreachable" to "network initializing" and goes through network initialization. This involves re-registering names and multicast groups on the new network.

Because applications vary widely in purpose one from another, they also vary widely in how they respond to changes in network connectivity. Some specific examples are given below.

An Internet browser application may query the new network configuration received from the DHCP server to see if the network contains a proxy server. If so, the browser can set itself up to use the proxy server.

The DNS Resolver service caches DNS server addresses so it is important that it notes any changes in those addresses. The new network configuration received from the DHCP server has this information.

The NetBIOS client notes changes to the WINS server addresses (also found in the new network's DHCP configuration) and uses those new WINS servers for NetBIOS name registrations, releases, and queries.

The printer service notes changes to the printer addresses (also found in the new network's DHCP configuration) and uses the new printers for servicing print requests.

An application program that receives a multicast stream via the Scalable Reliable Multicast mechanism may check to see if the available bandwidth on the new network connection is lower than it was on the previous network connection. If so, the application may inform the sending application so that it can throttle the send rate and thus avoid data loss within the network and subsequent retransmissions. In a like manner, an application sending a multicast stream may check the availability of bandwidth on the new network connection and adjust its send rate accordingly.

While the computing device was not connected to the network, a logon service may have granted access to a user on the basis of cached credentials. On reestablishing

the network connection, the logon service can query a security server for the logon credentials.

V. The NLRSP: Providing Network Connectivity Information

The section immediately preceding this one gives examples of applications' responses to network media connect events. Because applications may need to adapt their behavior to the nature of the logical networks to which their host computer is attached, the applications seek to discover the nature of a newly connected network by communicating with devices on that network. Complications arise because networks differ in the set of discovery techniques applicable to them. Accordingly, one aspect of the present invention removes the burden of supporting all of the various discovery techniques from the applications. In accordance with the present invention, a set of services, for the sake of discussion called the Network Location Resolution Service Provider (NLRSP), is provided by the host computer to discover aspects of the new network connections. The discovered information is provided to applications through a common API.

Figure 5 shows the steps followed by one embodiment of the NLRSP invention. In Step 500, the NLRSP contacts the drivers of the network interfaces on the computer to discover which interfaces are currently connected to networks. In some embodiments, the NLRSP simply polls all the network drivers for this information. In other embodiments, the NLRSP registers with the drivers to be automatically notified whenever the driver senses a network connect event.

In step 502 of Figure 5, the NLRSP constructs names for each logical network connected to the computer. The name given to a logical network is created by the NLRSP in such a manner that the combination of the name and the method through which the name was obtained uniquely identifies the logical network within the context of the computer. The NLRSP also tries to generate names in such a manner that there is a "one-to-one" mapping between the names and the logical networks. This one-to-one mapping has two fundamental properties. First, different logical networks are given different names. Second, a given logical network is given the same name whenever the computer connects to it. These properties are called "uniqueness" and "persistence," respectively.

Because the network names are unique and persistent, applications may rely on them when selecting a configuration to use with a given logical network.

In order to try to construct a name that is both unique and persistent, the NLRSP follows a set formula for applying information it discovers about the network. First, the NLRSP searches the logical network for information, and only when it has enough information about the network does it construct a name that applies to no other network (uniqueness). Information used by embodiments of the NLRSP to construct network names includes the domain name of the network (obtainable via DHCP), static information entered by a user to name a network, and subnet addresses. On networks that support the 802.1X authentication protocol, the NLRSP can construct a network name based on the network identity string returned in an EAP Identity Request message. Second, by following a set formula, the NLRSP usually constructs the same name for the same logical network (persistence). Different embodiments of the invention may depend upon different methods for gathering logical network information, and some of these methods may, in certain circumstances, lead to names that violate either the uniqueness or the persistence property of an ideal mapping. The text accompanying Figures 9, 10A, and 10B explain some of these situations and show why the information provided by the invention is still useful.

Step 504 of Figure 5 illustrates that in some embodiments of the present invention, the NLRSP correlates the network name it constructs with other information. A globally unique identifier (GUID) may be given that identifies the computer's physical network interface through which the network is accessible. The GUID can be used by the application when asking for more information about the interface. The network name is also correlated to the APIs of the transport protocols supported by the network. Other information correlated to the network name may include the type of the network interface (e.g., Ethernet or Point-to-Point Protocol), the speed of the network interface, the name of the network access device which supports the network interface, and the port on the network access device through which this connection to the network is made. If the computer has multiple interfaces to the same logical network, then the NLRSP may generate information about each interface, and, while the network name for each interface will be the same, the different interfaces can be distinguished by their GUIDs.

In Step 506 of Figure 5, the NLRSP presents its information to applications. Applications ask for information and may register to be informed when the information changes. The NLRSP keeps track of the information given to an application and notifies the application whenever any of the information given changes or when new information becomes available.

Figures 6A and 6B illustrate the above points. Figure 6A shows a laptop PC 100 and three locations where the laptop's owner may wish to use it. In each of the three locations, the user connects the laptop to a logical network and the NLRSP discovers information about the logical network and creates a name for it. When the user is at home, her laptop communicates with the Internet via a dial-up link to a local Internet service provider (ISP). The NLRSP names this logical network "myISP.net." At the user's desk at work, the laptop connects to the corporate LAN 106. The NLRSP queries servers on the LAN and names that logical network "worldWideWork.com." Finally, the user takes her laptop when she volunteers at a local agency. The NLRSP names this logical network "helpingOut.org." It is important to realize that the specific name given by the NLRSP need not be intrinsically descriptive. Applications can rely on the name as long as it is unique and persistent.

Figure 6B shows, for each of the three locations of Figure 6A, the network information 600 provided by the NLRSP. The network name 602 created by the NLRSP is discussed in relation to Figure 6A. The NLRSP identifies the interface 604 through which the logical network can be accessed. Here, that identifier is the GUID of the hardware that supports the computer's physical interface to the logical network. In some embodiments, the NLRSP correlates the network name with other information that applications frequently need. The information provided by the NLRSP in Figure 6B includes the type of the interface 606 and the nominal speed 608 supported by that interface. If an application needs even more information, it can use the interface identifier 604 to ask for it.

Figure 6B also shows the configuration a particular application may choose when the laptop 100 is connected to a logical network at a given location. Here, the application uses one configuration 610 for both home and volunteer agency network connections and a separate configuration 612 when at work. The NLRSP does not force the application to

use a different configuration for each different network connection; it simply enables the application to do so, at the user's discretion.

Figures 7A and 7B illustrate a complication to the scheme identified above. Here the laptop PC 100 is at the user's home and the NLRSP has named the logical network connection "myISP.net." The user uses that connection to log into work, in the process setting up a virtual private network connection to the network at her work location. In some embodiments of the NLRSP, a second network name is added to the first. The NLRSP creates "worldWideWork.com" to describe a virtual, rather than a physical, network connection. This is shown by the interface type 606 of the new connection, "PPP," which indicates a point-to-point virtual connection. In order to choose the configuration best adapted to this connectivity situation, an application consults both network names created by the NLRSP.

Figures 8A through 8C illustrate a different complication, this one due to the networking technology called Network Address Translator (NAT). NAT allows several computing devices connected in one location to share a single connection to the Internet. In Figure 8A, the user comes home and sets up a connection to work but she need not dial-up myISP.net as she did in the scenarios of Figures 6A and 7A. Instead, she connects the laptop PC 100 to her home LAN 106 and shares the Internet connection already established by the server computing device 800. This is a complication because a straightforward embodiment of the NLRSP running on the laptop might produce the information shown in Figure 8B. That information, accurate as far as it goes, does not indicate the dial-up connection which is the most likely source of network troubles and bottlenecks. In Figure 8C, a fuller embodiment of the NLRSP complements the information shown in Figure 8B with information about the dial-up connection. Applications using the information provided in Figure 8C can get a more complete picture of the connectivity scenario in which they are working and can respond accordingly.

In certain circumstances, it may be very difficult to create a mapping that conforms to the strict requirements of uniqueness and persistence. In Figure 9, as in Figure 6A, the ISP provides the same network name to all its dial-in customers. If a user and her neighbor subscribe to the same ISP, then a laptop 100 will report the same

network name ("myISP.net") whether it is connected at the user's home or at her neighbor's house. This mapping may violate the uniqueness property because it uses the same network name for connections originating from two different locations. However, this mapping may still be useful in choosing which network configuration to use because the user would most likely want to use the same network configuration when dialing into her ISP regardless of her location when dialing.

In the scenario depicted in Figures 10A and 10B, the property of persistence is violated. One night, a system administrator changes the name of the corporate network from "worldWideWork.com" to "captivatingCareer.com." When the user plugs in her laptop 100 at work the next day, the invention maps the corporate network to the name "captivatingCareer.com." The mapping is not persistent because the same network is given different names on different days. Although the laptop has no configuration stored for use with the new name, this problem is easily remedied. The laptop presents the new network name to the user and asks for guidance as to which configuration to use. Realizing what happened, the user directs her laptop to use the configuration once associated with "worldWideWork.com."

Figures 11A, 11B, 11C, and 11D illustrate another type of information gathered by the NLRSP about the networks connected to the host computer. This information relates primarily to the other devices connected to the network. A network may be of one of four "connectivity types" and Figure 11A is a network diagram showing a computing device 100 connected to four different types of networks. (It is unlikely that one device would be connected to one network of each connectivity type, but it is not impossible, and this configuration is useful for illustrative purposes.) The NLRSP probes the networks for information and uses an algorithm such as the one portrayed in Figure 11B to determine the connectivity type of each network. The specifics of the Figure 11B algorithm are pertinent to the IP protocol, but the general method is widely applicable. Network 1100 is called an "ad hoc" network, defined to be a network not connected to any other network. The NLRSP determines this in step 1110 by noting that the host computing device's address on this network is in the private IP address range 169.254.0.0 through 169.254.255.255 and by noting that there is no gateway on the network. Network 1102 is a "managed" network, determined in step 1112 by the host 100 having a valid

address (not 0.0.0.0) not in the private domain range of an ad hoc network, the presence in the host's configuration for this network connection of a DNS server with address other than 127.0.0.1, and the host having a domain configured. The NLRSP contacts the DNS server 1108 to see that it exists. Network 1104 is called "unmanaged" because it is on the private side of a NAT server 800. (For a description of NAT, see the text accompanying Figure 8A). The NLRSP determines that this is an unmanaged network in step 1114 because it receives an Internet Connection Service (ICS) beacon periodically broadcast by the NAT server. (As a practical matter, it may take so long to detect the beacon that the NLRSP returns the connectivity type before detecting it.) Finally, network 1106 is of connectivity type "unknown" because the NLRSP's tests in steps 1110, 1112, and 1114 are not determinative of any of the three previous network types. A declaration of type "unknown" also results whenever the interface is not connected to a network because this algorithm depends upon information gathered from the network. When the interface becomes connected to a network, the algorithm is re-run to determine the connectivity type of the new connection.

In addition to determining the connectivity type of the networks, the NLRSP checks to see if each network provides connectivity to the Internet. Figure 11C shows how this may be accomplished. In step 1116, the NLRSP checks some preliminary information. Then, in step 1118, the NLRSP tries to resolve an Internet name. The particular name used is of little importance. If the name is resolved, then the network has connectivity to the Internet.

Figure 11D is a block diagram showing network connectivity type and Internet connectivity information for the four networks of Figure 11A. Note that by definition a network with connectivity type "ad hoc" does not have Internet connectivity. The network connectivity type and Internet connectivity information are returned for each network as optional fields added on to the block 600, first shown in Figure 6B.

In some circumstances, domain administrators need to set up group policies to tailor network usage. These policies are preferably based on connectivity information presented by the NLRSP. For example, an ICS group policy may allow users to enable and run ICS at home but prohibit them from using ICS if the NLRSP detects that the computer is connected to a corporate network. Similarly, for safety's sake, a user may run

a firewall application to guard the computer's network links if the user is at home or on the road. However, that firewall application becomes merely a nuisance if it is run on links to a network already protected by a dedicated firewall, such as a corporate network. The group policy may disable the firewall for those links, and only for those links, connected to the corporate network. Many other scenarios can be imagined where group policies are based on knowing the specific networks to which a computer is connected.

VI. An NLRSP API According to One Aspect of the Invention

This section details one embodiment of the present invention, directed toward Microsoft's "WINDOWS" operating system. In this embodiment, the invention is presented as an API that allows applications to enumerate all of the logical networks currently accessible from the application's host computer, to identify the name given to a logical network, to identify the transport address of the physical network interface to a logical network, and to identify the physical location of the network interface. In addition, the WSANSPIoctl function is used to notify applications when information previously provided to them changes.

WSANSPIoctl

The Windows Sockets WSANSPIoctl function provides a method for making Input/Output control calls to a registered namespace. It has the following structure:

```
INT WSAPI WSANSPIoctl
(
    HANDLE              hLookup,
    DWORD               dwControlCode,
    PVOID               pvInBuffer,
    PDWORD              pcbInBuffer,
    PVOID               pvOutBuffer,
    PDWORD              pcbOutBuffer,
    PWSACOMPLETION      pCompletion
);
```

Parameters:

hLookup

[in] Lookup handle returned from a call to WSALookupServiceBegin.

dwControlCode

[in] The control code of the operation to perform.

pvInBuffer

[in] A pointer to the input buffer for the operation.

cbInBuffer

[in/out] The size of the input buffer for the operation.

pvOutBuffer

[out] A pointer to the output buffer for the operation.

pcbOutBuffer

[in/out] A pointer to an integral value for the size of the output buffer.

pCompletion

[in] A pointer to a WSACOMPLETION structure.

Return Value:

Upon successful completion, WSANSPIoctl returns NO_ERROR. Otherwise, a value of SOCKET_ERROR is returned, and a specific error code can be retrieved by calling WSAGetLastError.

Error Codes:

WSANOTINITIALISED

A successful call to NSPStartup was not performed.

WSA_INVALID_HANDLE

hLookup was not a valid query handle returned by WSALookupServiceBegin, or a handle passed in *pCompletion* was invalid.

WSAEFAULT

The *pvInBuffer*, *pcbInBuffer*, *pvOutBuffer*, *pcbOutBuffer*, or *pCompletion* argument is not totally contained in a valid part of the user address space. Alternatively, the *cbInBuffer* or *cbOutBuffer* argument is too small, and the argument is modified to reflect the required allocation size.

WSAEOPNOTSUPP

The specified *dwControlCode* is unrecognized.

WSAEINVAL

A supplied parameter is not acceptable, or the operation inappropriately returns results from multiple namespaces when it does not make sense for the specified operation.

WSA_NOT_ENOUGH_MEMORY

There were insufficient resources to perform the requested operation.

WSAEWOULDBLOCK

A non-blocking (polling) request was issued and the desired condition was unsatisfied. If *dwControlCode* was set to SIO_NSP_NOTIFY_CHANGE, a polling change-notification request was made and nothing about the resulting data has changed.

WSA_OPERATION_ABORTED

A blocking request was unable to complete.

WSASYSCALLFAILURE

An APC completion was indicated in *pCompletion* and the operation completed immediately, but the system failed to queue a thread to complete the APC.

Remarks:

The WSANSPIoctl function is used to set or retrieve operating parameters associated with a namespace query handle.

Any IOCTL may block indefinitely, depending upon the relevant namespace's implementation. If an application cannot tolerate blocking in a WSANSPIoctl call, it uses overlapped Input/Output. For these operations, which cannot be completed immediately, completion is indicated later through the mechanism specified in the *pCompletion* parameter, which is a pointer to a WSACOMPLETION structure. If *pCompletion* is NULL, this is a blocking call. To make this call non-blocking and return immediately, set WSACOMPLETION::*Type* to LUP_NOTIFY_IMMEDIATELY.

WSACOMPLETION

```
typedef struct _wsacompletion
{
    enum _type
    {
        LUP_NOTIFY_IMMEDIATELY,
        LUP_NOTIFY_HWND,
        LUP_NOTIFY_EVENT,
        LUP_NOTIFY_PORT,
        LUP_NOTIFY_APC
    } Type;
    PVOID               Recipient;
    UINT_PTR            Completion;
    PWSAOVERLAPPED      Overlapped;
}
```

Members:

Type

Indicates the type of *Recipient.*

Recipient

Either an HWND, HANDLE to an event or completion port, or function address for an asynchronous procedure call.

Completion

For *Type* LUP_NOTIFY_HWND, this is the window message identifier to send. For *Type* LUP_NOTIFY_PORT, this is the completion key to use.

Overlapped

Used in overlapped operations.

The following IOCTL code (command) is supported by NLA:

SIO_NSP_NOTIFY_CHANGE

This operation checks if the query results returned via calls to WSALookupServiceBegin and WSALookupServiceNext remain valid. If *pCompletion* is NULL, this operation is a poll and returns immediately. If the query set remains valid, WSAEWOULDBLOCK is returned to indicate that later invalidation will require an asynchronous notification. If the query set has changed and is invalid, NO_ERROR is returned indicating success in polling for invalidation of the query set. Not all name resolution protocols will be able to

support this feature and thus this call may fail with WSAEOPNOTSUPP. A query containing data from multiple providers cannot call this IOCTL and will return WSAEINVAL.

pvInBuffer, *pcbInBuffer*, *pvOutBuffer*, and *pcbOutBuffer* are ignored.

Some protocols may simply cache the information locally and invalidate it after some time, in which case a notification is issued to indicate that the local cache has been invalidated.

For name resolution protocols where changes are infrequent, it is possible for a namespace service provider to indicate a global change event that may not be applicable to the query on which change notification was requested and issued.

Immediate poll operations are usually much less expensive because they do not require a notification object. In most cases, this is implemented as a simple Boolean variable check. Asynchronous notification, on the other hand, in addition to expenses related to the notification object involved with signaling the change event, may (depending on the implementation of the namespace service provider) necessitate the creation of dedicated worker threads or inter-process communication channels.

To cancel an asynchronous notification request, simply end the original query with a WSALookupServiceEnd call on the affected query handle. Canceling asynchronous notification for LUP_NOTIFY_HWND will not post any message, however an overlapped operation will be completed, and notification will be delivered with the error WSA_OPERATION_ABORTED.

Upon signaling an invalidation of query data, a namespace should permit, through extended semantics of WSALookupServiceNext, an application to query what about the data has changed.

Queries

Queries are performed using the following namespace calls.

NSPLookupServiceBegin

The returned *lphHandle* for the query is an internal LPNLA_QUERY_HANDLE structure allocated on a private heap for the namespace. It is treated as an opaque object by the application and not modified. It

is not a true handle so WSAGetOverlappedResult cannot be used. The resulting query set comprises the adapter enumeration from GetAdaptersInfo and a registry-merge of non-active, saved network names. Registry-merge is the process of obtaining user-specific, persistent network names from HKEY_CURRENT_USER, combining the resulting list with global-system, persistent network names from HKEY_LOCAL_MACHINE, matching or adding them as appropriate to the adapter enumeration list, and producing a list of WSAQUERYSETW structures.

LUP_RETURN_NAME, LUP_RETURN_COMMENT, LUP_DEEP, and LUP_RETURN_BLOB may be set in *dwControlFlags*. These affect the output from future calls to NSPLookupServiceNext. Future calls only return the networks that contain the requested fields. For example, if LUP_RETURN_BLOB is requested, then only those networks with blob (binary large object) information will be returned from calls to NSPLookupServiceNext regardless of which controls flags are passed in. When LUP_DEEP is specified, NLA returns extended network information which may take a long time to acquire.

Error Codes:

WSANOTINITIALISED

A successful call to NSPStartup was not performed.

WSASERVICE_NOT_FOUND

lpProviderId or *lpqsRestrictions->lpServiceClassId* was an invalid GUID.

WSAEINVAL

A parameter was not validated, or *lpqsRestrictions->lpafpProtocols* contains something other than IP-based protocols, or *dwControlFlags* contains an invalid flag. Alternatively, network or friendly name filtering was indicated by *lpqsRestrictions->lpszServiceInstanceName*, or there was a non-NULL *lpqsRestrictions->lpszComment*, and LUP_RETURN_NAME or LUP_RETURN_COMMENT was not specified in *dwControlFlags*, respectively.

WSANO_DATA

LUP_CONTAINERS was specified in *dwControlFlags*.

WSAEFAULT

An access violation occurred while reading from or writing to user-supplied parameters.

WSA_NOT_ENOUGH_MEMORY

The system was unable to allocate a query handle or share memory from a system service.

WSAEACCESS

The calling thread lacked security permissions to access the user-persistent networks.

WSAESYSNOTREADY

The NLA system service was not available.

NSPLookupServiceNext

First, the *lphHandle* is checked to see that it is a true handle returned by NSPLookupServiceBegin. If valid, a new WSAQUERYSETW is copied into *lpqsResults*. If LUP_RETURN_BLOB was specified and various information about the networks exists, the information is returned in lpqsResults->lpBlob. Pointers in the view of the mapped file are offset relative to the beginning of each individual WSAQUERYSETW. These are changed to actual addresses in the namespace's process address space before being returned from this call. The resulting WSAQUERYSETW is formatted as follows.

LUP_RETURN_NAME, LUP_RETURN_COMMENT, and LUP_RETURN_BLOB may be set in *dwControlFlags*. These are independent of those supported in NSPLookupServiceBegin.

Upon receiving a WSA_E_NO_MORE from a WSALookupServiceNext call, if WSANSPIoctl was called with SIO_NSP_NOTIFY_CHANGE which succeeded immediately or returned pending, then WSALookupServiceNext may be called again to enumerate those networks that have changed. *dwOutputFlags* will contain one of the following:

RESULT_IS_ADDED

RESULT_IS_CHANGED

RESULT_IS_DELETED

A change can be indicated when any field which was requested at the time of WSALookupServiceBegin changes in any way.

When the enumeration completes, WSA_E_NO_MORE is received. A SIO_NSP_NOTIFY_CHANGE may be reissued at any time.

Error Codes:

WSANOTINITIALISED

A successful call to NSPStartup was not performed.

WSAEINVAL

lpdwBufferLength was NULL, *dwControlFlags* has an invalid value, or LUP_FLUSHPREVIOUS was specified when there was nothing to flush.

WSA_INVALID_HANDLE

hLookup is not a valid query handle returned by NSPLookupServiceBegin.

WSAESYSNOTAVAIL

The NLA system service is not available.

WSAEACCESS

The calling thread lacked security permissions to access the user-persistent networks.

WSAEFAULT

lpdwBufferLength indicated a buffer in *lpqsResults* too small to hold the resulting WSAQUERYSETW. The size required is returned in *lpdwBufferLength*. If the application is unable to supply a buffer of the specified size, it calls WSALookupServiceNext with *dwControlFlags* set to LUP_FLUSHPREVIOUS to skip over the entry. Alternatively, an access violation occurred while writing data to the buffer supplied in *lpqsResults*.

WSA_NOT_ENOUGH_MEMORY

The namespace is unable to acquire network information from the NLA system service due to a lack of memory in the calling process.

WSA_E_NO_MORE

No more networks are left to enumerate in the query result.

WSAESYSNOTREADY

The NLA system service is not available.

NSPLookupServiceEnd

The lookup handle is deallocated from the namespace private heap, and if the reference count hits zero, the LPC connection to the service is closed.

Error Codes:

WSANOTINITIALISED

A successful call to NSPStartup was not performed.

WSA_INVALID_HANDLE

hLookup was not a valid query handle returned by NSPLookupServiceBegin.

NLA_BLOB

When LUP_RETURN_BLOB is specified for a given query, NLA attaches relevant NLA_BLOB entries to the resultant data in calls to NSPLookupServiceNext via (PNLA_BLOB)*lpqsResults->lpBlob->pBlobData* with the following format:

```
typedef enum _NLA_BLOB_DATA_TYPE
{
    NLA_RAW_DATA                = 0,
    NLA_INTERFACE               = 1,
    NLA_802_1X_LOCATION         = 2,
    NLA_CONNECTIVITY            = 3,    // LUP_DEEP required.
    NLA_ICS                     = 4,    // LUP_DEEP required.
} NLA_BLOB_DATA_TYPE, *PNLA_BLOB_DATA_TYPE;

typedef enum _NLA_CONNECTIVITY_TYPE
{
    NLA_NETWORK_AD_HOC          = 0,
    NLA_NETWORK_MANAGED         = 1,
    NLA_NETWORK_UNMANAGED       = 2,
    NLA_NETWORK_UNKNOWN         = 3,
} NLA_CONNECTIVITY_TYPE, *PNLA_CONNECTIVITY_TYPE;

typedef enum _NLA_INTERNET
{
    NLA_INTERNET_UNKNOWN        = 0,
    NLA_INTERNET_NO             = 1,
    NLA_INTERNET_YES            = 2,
} NLA_INTERNET, *PNLA_INTERNET;
```

```
typedef struct _NLA_BLOB
{
    struct
    {
        NLA_BLOB_DATA_TYPE          type;
        DWORD                       dwSize;
        DWORD                       nextOffset;
    } header;

    union
    {
        // header.type -> NLA_RAW_DATA
        CHAR                        rawData[1];

        // header.type -> NLA_INTERFACE
        struct
        {
            DWORD                   dwType;
            DWORD                   dwSpeed;
            CHAR                    adapterName[1];
        } interfaceData;

        // header.type -> NLA_802_1X_LOCATION
        struct
        {
            CHAR                    information[1];
        } locationData;

        // header.type -> NLA_CONNECTIVITY
        struct
        {
            NLA_CONNECTIVITY_TYPE   type;
            NLA_INTERNET            internet;
        } connectivity;
```

```
        // header.type -> NLA_ICS
        struct
        {
            struct
            {
                DWORD                   speed;
                DWORD                   type;
                DWORD                   state;
                WCHAR                   machineName[256];
                WCHAR                   sharedAdapterName[256];
            } remote;
        } ICS;
    } data;
} NLA_BLOB, *PNLA_BLOB, *FAR LPNLA_BLOB;
```

Notification Mechanism

To eliminate the need for worker threads in the namespace dynamic library to wait upon change notification events and then translate them into the user-requested notification type, an NLA system service, running as localsystem under svchost.exe, is used. This also minimizes the number of recipients of change events to a single process for the entire system which then filters out the events only to the processes that are interested in such changes.

Upon system startup, the NLA service starts an LPC server port in a worker thread. It then blocks waiting for connection and notification-change registrations from a process using the namespace. When a new connection is registered, a FILE_MAP_READ file mapping handle is given to the calling client, and an LPC request is issued to obtain the network data separately.

When WSANSPIoctl is invoked upon a valid query handle with the operation SIO_NSP_NOTIFY_CHANGE, the service is given a duplicate of the query's registry key handle, and a change notification request is registered via the LPC connection. When the DHCP service signals in the service's main thread, the service reconstructs the file mapping, performs all client change-notifications, and clears the registration list. If a particular user registry key updates, that particular registration is notified and removed from the registration list. If an LPC connection is broken, the service removes that application's query sets from the registration set.

Superfluous notifications are permissible.

Network Persistence and Friendly Names

The mapping method maintains its quality of persistence even through the period during which a network is not active. That is to say, a network is given the same name whether or not that network is currently active. This property allows an application to configure to an inactive network, waiting for the network to become active.

NSPSetService

Parameters:

lpqsRegInfo

This WSAQUERYSETW is either from a query result or is manually constructed.

essOperation

Supported Operations:

RNRSERVICE_REGISTER

The supplied network WSAQUERYSETW from *lpqsRegInfo* is made persistent in the fashion indicated by *dwControlFlags*.

RNRSERVICE_DELETE

If the supplied network WSAQUERYSETW from *lpqsRegInfo* is persistent, then it will be removed.

dwControlFlags

The operation performs an *essOperation* with either or both of the following options:

NLA_FRIENDLY_NAME

The *lpszComment* field of the supplied network WSAQUERYSETW from *lpqsRegInfo* is valid and will be stored persistently. This cannot be specified with RNRSERVICE_DELETE, instead use RNRSERVICE_REGISTER with a NULL-terminated string to clear a friendly name.

NLA_ALLUSERS_NETWORK

The entry is stored persistently under HKEY_LOCAL_MACHINE, making it available during queries to all users on the local machine. To delete a network from under HKEY_CURRENT_USER, this flag must

not be specified. This flag is only valid in the security context of a local system administrator.

Error Codes:

WSANOTINITIALISED

A successful call to NSPStartup was not performed.

WSASERVICE_NOT_FOUND

lpProviderId indicated an invalid provider, or *lpqsRegInfo->lpServiceClassId* indicated a service not provided by this namespace.

WSAEINVAL

essOperation or *lpqsRegInfo* was invalid. The network name may be missing in *lpqsRegInfo->lpszServiceInstanceName*, or a friendly name may be missing in *lpqsRegInfo->lpszComment* when NLA_FRIENDLY_NAME is specified in *dwControlFlags* or when *lpqsRegInfo->lpVersion*, *lpqsRegInfo->lpBlob*, *lpqsRegInfo->lpszContext*, or *lpqsRegInfo->lpcsaBuffer* is non-NULL. Also, this error can be returned when RNRSERVICE_DELETE is requested in *essOperation* while specifying NLA_FRIENDLY_NAME in *dwControlFlags*. To clear a friendly name, use RNRSERVICE_REGISTER with a NULL-terminated string for the friendly name.

WSAEFAULT

An access-violation occurred while examining the supplied parameters.

WSAEACCESS

NLA_ALLUSERS_NETWORK was specified in *dwControlFlags* while not in the security context of a local system administrator, or the calling thread lacked the access to store the information for the current user, or RNRSERVICE_DELETE was specified for a persistent network which did not exist.

Application Termination

Because LPC is used, cleanup is simple as the NLA service will be notified of port closure.

VII. Providing Physical Location Information

The preceding sections describe aspects of the present invention as a service that discovers and presents to applications the identities of the logical networks to which their host computer is attached. While helpful in many scenarios, this information is sometimes not sufficient. Only with relatively precise geographic location information can some applications and services adequately conform their behavior to the realities of their location. For example, a desktop computer can use knowledge of its physical location in choosing the proxy servers to which it connects, in setting the time zone value for its clock, in choosing a nearby printer as the system default, and in choosing which virtual private networks to join. A server can use the locations of its remote clients to measure the geographic spread of interest in its services, to restrict content, and to assess shipping charges. Network administrators can use location information to enhance physical reliability of network links by ensuring that the links are diversely routed, that is, that the links do not share a physical path.

Accordingly, one aspect of the present invention collects into one service the various methods for discovering physical location information available on the network. The service decides which method or methods are applicable to each network interface on the computer, applies those methods, and presents the resulting location information.

In one embodiment, the service reports information about the quality of the location information discovered. Some location discovery methods do not report on the quality of their information and some methods report on the quality of their information but are incapable of providing precise location information. Other methods are able to provide precise location information but the owner of that information prevents it from doing so. In these cases, privacy may be at issue. Advertising the precise physical location of key equipment such as servers, core routers, or interconnection points may not be something a security-conscious corporation would want to do. However, even if this information were provided with a large error range to alleviate the security concerns, many applications may still find the information useful.

An application might want to know which method the service used to discover the location information and the range of errors associated with that method. This is especially true if more than one method is available on a given network interface. One

embodiment of the system of the present invention provides multiple location records, one produced using each method, and sorts them in the order of increasing error range.

On a given computer, the locations reported for all of its network interfaces will normally be the same. In some situations, however, the information may differ because the information provided by some methods (see below for more details) reflects the location of the network, rather than the location of the network interface on the computer. The difference arises because a computer could have, for reliability reasons, network links that travel different paths before connecting to a network in different locations. The location reported for different network interfaces on a given computer may also differ if the location records were generated using different methods of location discovery.

According to another aspect of the present invention, the services described above may be presented to applications via a common API. The location information, however collected, is converted into a common format and keyed to the names of the logical networks. In one embodiment, physical location is formatted as a latitude, longitude pair. In other embodiments, other formats are supported, some including the altitude above sea level. The API may also present quality information and the methods used by the service to discover the information. The API also allows for additional, optional fields the contents of which depend upon the discovery method used. In one embodiment, the system of the invention informs users of the API when information provided to them changes or when new information becomes available. An application may provide to the API a threshold so that location changes of a magnitude below the threshold are not reported by the API.

The aspect of the present invention that presents physical location information may be embodied as a separate service with its own API or may become another service provided by the NLRSP described in sections V and VI above. Figures 12A and 12B show an example of how physical location information can be reported by the NLRSP. The situation is the same as in Figures 7A and 7B, to wit, the user is at home, dialed into her ISP, and has set up a virtual private network link through her ISP to the network at her work location. The information returned by the NLRSP 1200 contains the fields previously described in item 600 of Figure 7B plus a physical location field 1202, a location method field 1204, and an error range field 1206. As in the scenario of Figures

7A and 7B, this embodiment gives the user two records and applications check both before deciding how to proceed. In particular, note that the first method returns the physical location of the user's home (based on a ZIP code she entered earlier) while the second method returns the location of a server located at her place of work, possibly continents away.

Figures 13A through 13D illustrate how one embodiment of the present invention can be practiced. The Figures break down the major functions performed by the physical location aspects of the NLRSP into constituent steps.

Figure 13A shows what the NLRSP does when the computer starts up. The NLRSP creates a list of those network interfaces that are connected to live networks. Proceeding through the list, the NLRSP consults a table that lists which physical location methods may be applicable to the interface. (Many of the methods are detailed in the following section.) The NLRSP runs each method and, if the method is successful, collects the information that the method provides. That information includes physical location and, depending upon the method, may include error information and other information. The information is converted, if necessary, into a standard format, such as, for example, location expressed as longitude and latitude. Records are created that contain this information along with an indication of the method used to obtain the information.

Some embodiments of the NLRSP may not perform the procedure of Figure 13A on startup. Instead, these embodiments wait until they receive a media connect event and then perform the steps of Figure 13D (see below), much to the same effect.

Figure 13B shows what this embodiment of the NLRSP does when an application or system service requests physical location information. The NLRSP reports the information it has collected about the interface. If there is information from more than one method, that information is passed to the requestor, sorted, if possible, in order of increasing error range. The NLRSP also records "what was reported to whom." That information is used in the procedure of Figure 13D.

In Figure 13C, an application or system service indicates to the NLRSP that it is interested in being notified when information changes. The application or system service can also specify a threshold: changes of a magnitude below that threshold are not of

interest and are not reported. In this embodiment, the information is stored in the "what was reported to whom" records mentioned with respect to Figure 13B.

The NLRSP performs the procedure of Figure 13D when it is informed of a network event. Some events, such as a change in the network data rate, are immaterial to the physical location aspects of the NLRSP and trigger no further action. Other events, such as media connect and disconnect, change in carrier strength, etc., may invalidate the information stored by the NLRSP. For these events, the NLRSP discards its previously collected physical location information. Then it runs the discovery procedure described above with respect to Figure 13A. Having updated its stored information, the NLRSP checks to see if it should inform any applications or system services of the change. It checks the "what was reported to whom" records and reports the new information if (1) the information actually changed, (2) the application or system service indicated (either explicitly or implicitly) interest in this information, and (3) the change is greater than the threshold set by the procedure of Figure 13C.

Other embodiments of the present invention may implement the functions of the NLRSP in a manner very different from the methods described in Figures 13A through 13D. These Figures are meant to be illustrative only and do not limit the scope of the present invention.

VIII. Various Techniques for Determining Physical Location of a Network Interface

This section presents some of the many ways that a service according to the present invention may discover physical location information. Different network interfaces may support different sets of these methods and the present invention may be embodied to use any and all of the methods supported. This diversity is especially useful because many of these methods were not designed for, and have not been modified to accommodate, use with the present invention. The quality of their information may vary widely.

As new methods for discovering physical location are implemented, embodiments of the present invention can adjust to accommodate them, without having to change the API through which the information is presented to applications.

Per-Interface Domain Name

One embodiment of the present invention supports the use of the per-interface DNS domain name as the location information for a network interface. This information is either obtained locally from a static configuration or from DHCP.

DHCP Option for Physical Location

An extension to the per-interface DNS domain name allows the DHCP server to send a DHCP option to its clients containing their physical location expressed in terms of longitude and latitude. The option format is the same as for RFC 1876 (see below) and can be used by Dynamic DNS without changing or understanding the format.

Subnet to Directory Service Location Attribute

Some directory services contain a physical location attribute for some objects. These objects include site and subnet objects. According to one aspect of the present invention, the IP addresses of the network interface are used to discover the location attribute of the subnet object.

DNS Records via RFCs 1876 and 1712

Internet Engineering Task Force RFCs (Requests for Comment) 1876 and 1712 describe two DNS record formats that contain physical location expressed in terms of longitude and latitude. RFC 1876 supports error ranges on the location information. The fields in these records are:

VERSION: The version number of the representation. Implementations are required to check this field and make no assumptions about the format of unrecognized versions.

SIZE: The diameter of a sphere enclosing the described entity, in centimeters, expressed as a pair of four-bit unsigned integers, each ranging from zero to nine, with the most significant four bits representing the base and the second number representing the power of ten by which to multiply the base. This allows sizes from 0e0 (< 1 cm) to 9e9 (90,000 km) to be expressed. Four-bit values greater than 9 are undefined, as are values with a base of zero and a non-zero exponent.

HORIZ PRE: The horizontal precision of the data, in centimeters, expressed using the same representation as SIZE. This is the diameter of the horizontal "circle of

error," rather than a "plus or minus" value. (This was chosen to match the interpretation of SIZE; to get a "plus or minus" value, divide by 2.)

VERT PRE: The vertical precision of the data, in centimeters, expressed using the same representation as for SIZE.

LATITUDE: The latitude of the center of the sphere described by the SIZE field, expressed as a 32-bit integer, most significant octet first (network standard byte order), in thousandths of a second of arc. 2^31 represents the equator; numbers above that are north latitude.

LONGITUDE: The longitude of the center of the sphere described by the SIZE field, expressed as a 32-bit integer, most significant octet first (network standard byte order), in thousandths of a second of arc, rounded away from the prime meridian. 2^31 represents the prime meridian; numbers above that are east longitude.

ALTITUDE: The altitude of the center of the sphere described by the SIZE field, expressed as a 32-bit integer, most significant octet first (network standard byte order). Altitude above (or below) sea level may be used as an approximation of altitude.

Names

Another source of information is the name space itself. The name of routers often includes some gross information, such as "Vienna" or "West Orange."

whois

RFC 954 describes the protocol used to communicate to the whois server. The server returns contact location which may not be the same as the physical location of the network. In addition, the data may be copyrighted which may preclude placing them in a database. Also, the format is highly informal, which means that a real-time client-based solution would require some artificial intelligence. However, it is possible to combine two sources of whois data, the network address registration maintained by ARIN, RIPE, and the APNIC, and the domain name registration, maintained by Network Solutions and by national registries. There are technical difficulties, dealing with a country-specific address formats, and logical difficulties as well because the information refers to the owner of the name or of the address block and not to the geographical location of the

computer. The correlation is better with addresses than with names, and better still when the address block is small.

NetGeo

CAIDA (Cooperative Association for Internet Data Analysis) is developing a database of Internet geographic information and a visual trace route program that uses the database. To find the latitude and longitude values for a domain name, NetGeo first searches for a record containing the target name in its own database. The NetGeo database caches the location information parsed from the results of previous whois lookups to minimize the load on whois servers. If a record for the target domain name is found in the database, NetGeo returns the requested location information. If no matching record is found in the NetGeo database, NetGeo performs one or more whois lookups using the InterNIC or RIPE whois servers until a whois record for the target domain name is found. NetGeo publishes a set of APIs to communicate to its servers but does not guarantee the availability of the service. The server rate limits the number of queries made from a particular IP address.

Radio Ranging

A computer connected to a wireless network can often approximate its distance to a radio access point by measuring the strength of the RAP's signal. If the computer can approximate its distance from a few RAPs, and if the geographical positions of the RAPs are known, then the computer can triangulate and estimate its location. RAP locations can be discovered either from the RAPs themselves, from a DNS location record, or from a directory service.

The accuracy of the triangulation method depends upon knowing several factors, such as the strength of the RAP's signal at the RAP and how that strength decreases with increasing distance from the RAP. The latter factor depends in turn upon the specifics of the configuration of the RAP and upon the obstacles between the RAP and the computer. Because a computer will generally not be able to derive these factors, the factors may be measured beforehand. In one method, each RAP measures the strength of the signal it receives from other RAPs within its range. Using the known physical locations of the RAPs, a signal strength vs. distance function can be derived for each RAP. This information can be put into a signal strength contour map and the computer can access

this map when it wants to determine its location. The accuracy of the map can be enhanced by measuring signal strengths at known locations other than the locations of the RAPs.

Traceroute

Traceroute can be used in a way similar to radio ranging. Several "traceroute" requests to the target address are made, if possible from different locations. Some idea of the location can inferred from the geographical location of the addresses in the path. Traceroute information can be combined with timing information to refine the location information.

TAPI

When a RAS connection is made, the IP connection's physical location may not be valid and some RAS information may be used. This information consists of the area code from which the call is being made and the telephone number that is being called.

Global Positioning System (GPS)—NMEA 0183

The method of discovering physical location information by querying a GPS receiver connected to the computer is different from the other methods described above because those methods discover information from the network connection. However, GPS information is important. Information available from the network can be used to improve the location. The NMEA 0183 data format varies from GPS unit to GPS unit because most manufacturers insert their own proprietary extensions to the format, but there are a few standard message sentences which are available on most units that claim NMEA compliance.

Differential GPS (DGPS) is an extension to GPS that increases the accuracy of physical location information from about 100 meters to about 5 meters. It deploys GPS stations at known locations. These stations work out the errors in the GPS signal that show up as inaccuracies of the GPS calculated position from their known position. The errors are sent to nearby GPS receivers that then remove the errors from their calculations. Errors are normally sent by radio but a new system (WAAS) being worked on by the FAA transmits the errors to airplanes via geo-stationary satellites. Error information is also available over the Internet.

In a variation on DGPS, estimated location and satellites in view are downloaded to a cellular telephone GPS system from a network. Location information from the GPS system is transmitted to network servers who use correction information from DGPS systems to calculate locations and pass the information on to applications.

802.1X Authentication Protocol

The 802.1X protocol is used by a client computer to authenticate itself to a network access device, typically a RAP or Ethernet switch. The client computer receives an Identity Request message which can contain information such as the identity of the network (as discussed in Section V above), an identifier of the access device, and the port number on the access device to which the computer is connected. If the client computer has access to a map of the topology of the network, then it can use this information to determine its location relative to the access device. If, in addition, the map gives the physical locations of the network access devices, then the client computer can determine its physical location as well.

Some embodiments of the NLRSP present the 802.1X information as another interface on the client computer.

The embodiments described above are illustrative only and are not intended to limit the scope of the present invention. In particular, the embodiments describe the invention with respect to TCP/IP networking technologies and with respect to Microsoft "WINDOWS" operating systems but the invention applies as well to other networking technologies and to other computer technologies. The invention applies to all networks, to wireless as well as to wired network technologies. Therefore, the invention as described herein contemplates all such embodiments as may come within the scope of the following claims and equivalents thereof.